APPLICATION FOR UNITED STATES PATENT

by

MICHAEL EDWARDS

for

SYSTEMS AND METHODS TO COMPARE OPERATIONAL STATUSES OF SWITCH FEATURE PACKAGES

SHAW PITTMAN LLP
1650 Tysons Blvd., 14th Floor
McLean, Virginia 22102-4859
(703) 770-7900

Attorney Docket No.: BS01-237

SYSTEMS AND METHODS TO COMPARE OPERATIONAL STATUSES OF SWITCH FEATURE PACKAGES

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] Embodiments of the present invention relate to configuration of telecommunications switches. More particularly, embodiments of the present invention relate to systems and methods for comparing operational statuses of switch feature packages.

BACKGROUND INFORMATION

[0002] Known telecommunications networks have switches that include software modules that support one or more features. For example, a switch can include software that supports features (e.g., feature packages) such as an Integrated Services Digital Network ("ISDN"), ISDN attendant services, modified calling line disconnect, ISDN multipoint Digital Subscriber Line ("DSL"), music on hold, national ISDN base, type 2A/2B cellular interface via Signaling System 7 ("SS7"), caller identification ("caller ID") multiline, call waiting deluxe, Centrex network, RingMaster™ distinctive ringing, pay phone compensation, 800 number services, service switching point, talking call waiting, and so on. A switch feature package may or may not be loaded on a switch (e.g., the switch may or may not have the capacity to loaded and/or execute the switch feature package). Even when a switch feature package is loaded, it may or may not be enabled based on service operation factors. For example, a feature package of a switch may not be enabled due to a present lack of demand for the service that corresponds to that feature package. After

demand for that service exists (or is anticipated based on, for example, a marketing campaign), the feature package can be enabled.

[0003] When a new switch is installed to replace an old switch (e.g., a switch upgrade), the status of the feature packages (e.g., enabled, disabled, and so on) in the new switch and the old switch should be the same. When features that were enabled in the old switch are not enabled in the new switch, customers may no longer receive services corresponding to the non-enabled features. For example, a customer may no longer receive Caller ID information, be able to utilize Call Waiting, and so on. A known method for comparing the operational status (e.g., enabled, disabled) of feature packages of two switches is a manual, line by line, comparison. In view of the foregoing, a substantial need exists for systems and methods that can compare the operational statuses of switch feature packages.

BRIEF SUMMARY OF THE INVENTION

[0004] Embodiments of the present invention relates to systems and methods for comparing feature package operational statuses of two or more switches. In an embodiment, the system includes a first switch and a second switch. The first switch includes a first set of feature packages, and the second switch includes a second set of feature packages. The system also includes a computer coupled to the first switch and to the second switch. The computer is to receive a first set of feature package information and a second set of feature package information. The first set of feature package information corresponds to the first set of feature packages, and the second set of feature package information corresponds to the second set of feature packages. The computer is to compare the first set of feature package information and the second set of feature package information.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Figure 1 is a schematic diagram showing an embodiment of the present invention.

[0006] Figure 2 illustrates an embodiment of the present invention.

[0007] Figures 3A through 3I illustrate another embodiment of the present invention.

[0008] Figure 4 illustrates another embodiment of the present invention.

[0009] Figure 5 is an illustration of a graphical user interface in accordance with an embodiment of the present invention.

[0010] Figure 6 is a flow diagram illustrating an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0011] According to an embodiment of the present invention, a system can retrieve a first set of feature packages operational status information from a first switch and a second set of feature packages operational status information from a second switch. The system can compare the first set of feature packages operational status information and the second set of feature packages operational status information. In an embodiment, the comparison can yield an output that indicates feature packages that are loaded in both the first switch and the second switch but do not have the same operational status. For example, a feature package can be loaded in both the switches, but enabled (e.g., active) in one switch and disabled (e.g., inactive) in the other switch. In another embodiment, the comparison can yield an output that indicates loaded feature packages and the operational status of each loaded feature package in each switch.

[0012] Figure 1 is a schematic diagram showing an embodiment of the present invention. A computer 101 is coupled to a first switch 110 via a communications link 105 and to a second switch 120 via a second communications link 106. As used to describe embodiments of the present invention, the term "coupled" encompasses a direct connection, an indirect connection, or a combination thereof. Moreover, two devices that are coupled can engage in direct communications, in indirect communications, or a combination thereof.

[0013] In an embodiment, computer 101 includes a processor and a memory. The processor can be, for example, an Intel Pentium® 4 processor, manufactured by Intel Corp. of Santa Clara, California. As another example, the processor can be an Application Specific Integrated Circuit (ASIC). The memory may be a random access memory (RAM), a dynamic RAM (DRAM), a static RAM (SRAM), a volatile memory, a non-volatile memory, a flash RAM, polymer ferroelectric RAM, Ovonics Unified Memory, magnetic RAM, a cache memory, a hard disk drive, a magnetic storage device, an optical storage device, a magneto-optical storage device, or a combination thereof. The memory of computer 101 can store a plurality of instructions adapted to be executed by the processor of computer 101. In an embodiment, computer 101 is coupled to switches 110 and 120 via network 102 and a network connection (e.g., data port, input/output port, etc.). Examples of network 102 include a Wide Area Network (WAN), a Local Area Network (LAN), the Internet, a wireless network, a wired network, a connection-oriented network, a packet network, an Internet Protocol (IP) network, at least a portion of the public switched telephone network ("PSTN"), or a combination thereof.

[0014] Each of switches 110 and 120 can include a plurality of feature packages. For example, first switch 110 can include feature package A 111, feature package B 112, . . . and feature package N 115. In an embodiment, second switch 120 can include a corresponding set of feature packages, such as a feature package A 121 corresponding to feature package A 111, a feature package B 122 corresponding to feature package B 112, . . . a feature package N 125 corresponding to feature package B 115. In another embodiment, second switch 120 can include additional feature packages, such as feature package X 127, that do not correspond to a feature package of first switch 110.

[0015] For example, first switch 110 can be a Lucent 5E3 switch (from Lucent Technologies Inc. of Murray Hill, New Jersey) and second switch 120 can be another Lucent 5E3 switch, and each of feature package A 111 and feature package B 121 can be a National ISDN Base feature package. As another example, first switch 110 can be a Nortel DMS-500 switch (from Nortel Networks Limited of Montreal, Quebec, Canada) of a first central office and second switch 120 can be another Nortel DMS-500 switch of a second central office, and each of feature package N 115 and feature package N 125 can be a CallerID with call waiting feature package. As a further example, the first switch and the second switch can be different types of switches (e.g., switches from different switch manufacturers) that include comparable sets of feature packages. In other embodiments of the present invention, first switch 110 can be a host switch and second switch 120 can be a remote switch of the host switch, or each of first switch 110 and second switch 120 can be remote switches (e.g., remotes

of the same host, remotes of different hosts, etc.), or each of first switch 110 and second switch 120 can be host switches, and so on.

[0016] In an embodiment, computer 101 can include instructions (e.g., a software script) to query each of first switch 110 and second switch 120 for feature package operational status information. For example, computer 101 can receive a first set of feature package operational status information from first switch 110 via a first Telnet session and can receive a second set of feature package operational status information from second switch 120 via a second Telnet session. Computer 101 can then compare the first set of feature package operational status information to the second set of feature package operational status information to determine relative differences, similarities, and/or matches in the sets of feature package operational status information.

[0017] Figure 2 illustrates an embodiment of the present invention. A data record 200 can include feature package operations status information that indicates differences in a first set of feature package operational status information from a first switch and a second set of feature package operational status information from a second switch. For example, data record 200 includes feature package operational status information for feature packages loaded in each of a first switch and a second switch where the operational status of corresponding feature packages is different, e.g., a feature package in one switch is enabled and a corresponding feature package in the other switch is not enabled.

[0018] Data record 200 can include a plurality of data entries such as data entry 210. Data entry 210 corresponds to a type of feature package loaded in each of the first

switch and the second switch (e.g., an Automatic Message Accounting ("AMA") for Type 2 Cellular feature package). Data entry 210 can include a plurality of data fields, where each data field can store a data value. In an embodiment, data entry 210 can include a feature package code field 212 to store a feature package code, a feature package name field 214 to store a feature package name, a feature package enabled in first switch field 216 to store a feature package enabled in first switch field value (e.g., a boolean value), and a feature package enabled in second switch field 218 to store a feature package enabled in second switch field value (e.g., a boolean value). In the embodiment of the present invention illustrated in Figure 2, each of the data entries in the data record 200 include information for feature packages having different operational statuses in the first switch and the second switch. For example, as shown by data entry 210, the AMA for Type 2 Cellular feature package is enabled in the first switch and disabled in the second switch.

[0019] Embodiments of the present invention, such as data record 200, can advantageously help to eliminate errors when a new switch is being set up and/or activated. For example, a central office may be converted from an old switch to a new switch. Typically, the new switch should have at least the same features as the old switch, e.g., so that customers do not lose one or more services when a cutover is made to the new switch from the old switch. For example, if the old switch had a CallerID feature package loaded and enabled and the new switch did not have a CallerID feature package loaded and enabled, there can be customer complaints and dissatisfaction when a cutover is made to the new switch and customers no longer receive CallerID information. Embodiments of the present invention can identify

differences in feature package operational status and thereby help bring new switches into an existing service environment and resolve problems from office conversions.

[0020] Figures 3A through 3I illustrate another embodiment of the present invention. A data record 300 can include feature package operations status information that indicates matches between a first set of feature package operational status information from a first switch and a second set of feature package operational status information from a second switch. For example, data record 300 includes feature package operational status information for feature packages loaded in each of a first switch and a second switch where the operational status of corresponding feature packages is the same, e.g., a feature package in one switch is enabled and a corresponding feature package in the other switch is likewise enabled.

[0021] Data record 300 can include a plurality of date entries such as data entry 310. Data entry 310 corresponds to a type of feature package loaded in each of the first switch and the second switch (e.g., a Caller ID – MultiLine feature package). Data entry 310 can include a plurality of data fields, where each data field can store a data value. In an embodiment, data entry 310 can include a feature package code field 312 to store a feature package code associated with a feature package, a feature package name field 314 to store a feature package name, a feature package enabled in first switch field 316 to store a feature package enabled in first switch field value (e.g., a boolean value), and a feature package enabled in second switch field 318 to store a feature package enabled in second switch field value (e.g., a boolean value).

[0022] A data record such as data record 300 can be based on a comparison between two switches to help determine what features may be needed for a third switch. For

example, a new remote switch may be being configured. Comparing the feature packages loaded and enabled in two switches (e.g., two switches of the same vendor of the new remote switch, two local switches, two distant switches, a host of the new remote switch and a current remote of the host of the new remote switch, etc.) can identify the set of feature packages that are likely needed in the new remote switch.

[0023] As another example, a data record such as data record 300 can assist in a quick reload of Products/Services Inventory Management System ("P/SIMS") information in the event that feature information is accidentally erased, intentionally erased, and so on. P/SIMS is a switch software tracking and management system that is used after the "Right-to-Use" license has been purchased from the switch vendor to activate certain software products and feature packages based upon a lists of the switch vendor's available services at the switch's current level of its generic operating system.. For example, in P/SIMS all features of a remote may be been deactivated (e.g., by error). Embodiments of the present invention can compare a host of the remote and another remote of the host to identify features in the remote that need to be activated. Alternatively, the feature package operational status information of the remote switch can be compared with the feature package operational status information of the host to identify the feature packages enabled in the host that need to be enabled in the remote.

[0024] Figure 4 illustrates another embodiment of the present invention. A data record 400 can include feature package operations status information that indicates differences between a first set of feature package operational status information from a first switch and a second set of feature package operational status information from

a second switch. For example, data record 400 includes feature package operational status information for feature packages loaded in each of the first switch and the second switch where the operational status of corresponding feature packages is different. Data record 400 can include a plurality of date entries such as data entry 410. Data entry 410 corresponds to a type of feature package loaded in each of the first switch and the second switch (e.g., an Application Service Provider ("ASP") Network Busy Event Remote Termination Unit ("RTU") feature package). Data entry 410 can include a plurality of data fields, where each data field can store a data value. In an embodiment, data entry 410 can include a feature package code field 412 to store a feature package code associated with a feature package, a feature package name field 414 to store a feature package name, a feature package enabled in first switch field 416 to store a feature package enabled in first switch field value (e.g., a boolean value), and a feature package enabled in second switch field 418 to store a feature package enabled in second switch field value (e.g., a boolean value).

[0025] Figure 5 is an illustration of a graphical user interface ("GUI") in accordance with an embodiment of the present invention. A GUI 500 can prompt a user for information that specifies parameters of a feature package operational status information comparison. GUI 500 can include three comparison option boxes such as, for example, a same features but opposite status option box 511, a same features and same status option box 512, and a differences between generics but not same features box 513. In an embodiment of the present invention, feature package software modules are part of a generic software load of a switch, which can be purchased through a right-to-use license on an individual basis. A user can select one

of option boxes 511-513 to specify a type of comparison to be performed between two switches. GUI 500 can also include two or more switch identifier input fields such as, for example, first switch identifier field 521 and second switch identifier field 522. For example, a switch identifier can be a Common Language Location Identifier ("CLLI") code. In an embodiment, a user can also specify whether one or both of the identified switches are part of an office conversion via office conversion indicator fields 531 and 532.

[0026] Figure 6 is a flow diagram illustrating an embodiment of the present invention. A user can initiate (e.g., via a computer) a feature package comparison script (e.g., a software routine) (step 605). The user can select a type of comparison to be performed between two switches, e.g., a comparison to show features loaded in each switch that have different statuses, a comparison to show features loaded in each switch that have the same status, and a comparison to show features that are not loaded in both switches (step 610). The user can enter switch identifiers of a first switch and a second switch (step 615). Whether there are any future central office equipment orders involved with a central office associated with the first and/or second switch can be determined (decision 620). When there is a future central office equipment order involved with the first and/or second switch, the switch identifiers can be verified to determine whether they are valid (decision 635). When there is a future central office equipment order involved with the first and/or second switch, an office conversion incomplete status flag can be set (step 625), the user can enter a future telecommunications equipment order ("TEO") identifier of the central office

equipment order (step 630), and the switch identifiers can be verified to determine whether they are valid (decision 635).

[0027] When one or more of the switch identifiers are not valid, the user can re-enter one or more switch identifiers (step 615). When the switch identifiers are valid, a first set of feature package operational status information can be queried for and received (step 640) and a second set of feature package operational status information can be queried for and received (step 645). The selected type of comparison between the first set and the second set of feature package operational status information can be performed.

[0028] Embodiments of the present invention relate to data communications via one or more networks. The data communications can be carried by one or more communications channels of the one or more networks. A network can include wired communication links (e.g., coaxial cable, copper wires, optical fibers, a combination thereof, and so on), wireless communication links (e.g., satellite communication links, terrestrial wireless communication links, satellite-to-terrestrial communication links, a combination thereof, and so on), or a combination thereof. A communications link can include one or more communications channels, where a communications channel carries communications. For example, a communications link can include multiplexed communications channels, such as time division multiplexing ("TDM") channels, frequency division multiplexing ("FDM") channels, code division multiplexing ("CDM") channels, wave division multiplexing ("WDM") channels, a combination thereof, and so on.

[0029] In accordance with an embodiment of the present invention, instructions adapted to be executed by a processor to perform a method are stored on a computer-readable medium. The computer-readable medium can be a device that stores digital information. For example, a computer-readable medium includes a compact disc read-only memory (CD-ROM) as is known in the art for storing software. The computer-readable medium is accessed by a processor suitable for executing instructions adapted to be executed. The terms "instructions adapted to be executed" and "instructions to be executed" are meant to encompass any instructions that are ready to be executed in their present form (e.g., machine code) by a processor, or require further manipulation (e.g., compilation, decryption, or provided with an access code, etc.) to be ready to be executed by a processor. Appendix A contains an example of source code that can be used, among other things, to compare feature package operational status information of a first switch and a second switch.

[0030] Systems and methods in accordance with an embodiment of the present invention disclosed herein can advantageously compare feature package operational status information of two or more switches. Feature packages loaded in two switches but having different operational statuses can be identified. Feature packages loaded in two switches have the same operational status can also be identified. Embodiments of the present invention can advantageously identify feature package inconsistencies when switches are upgraded, replaced, reloaded, installed, and so on.

[0031] Embodiments of systems and methods to compare operational statuses of switch feature packages have been described. In the foregoing description, for purposes of explanation, numerous specific details are set forth to provide a thorough

understanding of the present invention. It will be appreciated, however, by one skilled in the art that the present invention may be practiced without these specific details. In other instances, structures and devices are shown in block diagram form. Furthermore, one skilled in the art can readily appreciate that the specific sequences in which methods are presented and performed are illustrative and it is contemplated that the sequences can be varied and still remain within the spirit and scope of the present invention.

[0032] In the foregoing detailed description, systems and methods in accordance with embodiments of the present invention have been described with reference to specific exemplary embodiments. Accordingly, the present specification and figures are to be regarded as illustrative rather than restrictive.

APPENDIX A

```
Dim Action As String                    'Determine how offices are compared
Dim clli1 As String                           '1st clli input for comparison
Dim clli2 As String                           '2nd clli input for comparison
Dim Ans1 As String                            'office clli1 job closed or not
Dim Ans2 As String                            'office clli2 job closed or not
Dim StrArr1(1) As String                      'choice of yes or no
Dim StrArr2(1) As String                      'choice of yes or no
Dim StrArr3(2) As String                      'choice of Action to perform for
comparision
Dim project1 As String                  'get clli1 project number if needed
Dim project2 As String                  'get clli2 project number if needed
Dim finding As String                   'read the screen
Dim tempDataArr1(999) As String         'all the software packages
Dim compare1(999) As String             'grab software feature first time
Dim answer1(999) As String              'yes or no of feature package
Dim page1(999) As String                'next page or end of page
Dim tempDataArr2(999) As String         'all the software packages
Dim compare2(999) As String             'grab software feature second time
Dim answer2(999) As String              'yes or no of feature package
Dim page2(999) As String                'next page or end of page
Dim match As String                     'match exists between offices for generic
                                             difference comparision
Dim skip As String                            'set up a skip variable to save
time
Dim row As Integer                            'which row of the screen is the
script looking at PSIMS info
Dim Command As String                   'Windows command to open Notepad.exe
Dim cllititle As String                 'Title column for table

Sub Main
      Syshide "Timer"
      skip = "no"
      row = 15
TryAgain:
      DlgStart
            DlgTitle "Compare 2 different offices of the same vendor"
            DlgItemType "Please click the bar you want and enter the information
                 as needed."
            StrArr3(0) = "Same features but opposite status"
            StrArr3(1) = "Same features and same status"
            StrArr3(2) = "Differences between generics but not same features"
            DlgItemButton  "Compare How?", Action, 4, StrArr3
            DlgItemAsk "1st CLLI", clli1, 0, 11
            StrArr1(0) = "Yes"
            StrArr1(1) = "No"
            DlgItemRadio "If necessary answer question, is the current office
                 conversion complete?", Ans1, 2, StrArr1
            DlgItemAsk "2nd CLLI", clli2, 1, 11
            StrArr2(0) = "Yes"
            StrArr2(1) = "No"
            DlgItemRadio "If necessary answer question, is the current office
                 conversion complete?", Ans2, 3, StrArr2
            DlgInquire
            DlgGetItemValue 0, clli1
```

```
            DlgGetItemValue 1, clli2
            DlgGetItemValue 2, Ans1
            DlgGetItemValue 3, Ans2
            DlgGetItemValue 4, Action
      DlgEnd

      StrLen% = Len(clli1)
      If StrLen% < 11 Then
            MsgType "1st CLLI is inaccurate.  Try Again"
            GoTo TryAgain
      End If
      StrLen% = Len(clli2)
      If StrLen% < 11 Then
            MsgType "2nd CLLI is inaccurate.  Try Again"
            GoTo TryAgain
      End If
Try2:
      project1 = ""
      If Ans1 = "No" Then
            MsgAsk  "Please input project number for 1st CLLI (Press Enter when
                  finished):", project1, 10
            StrLen% = Len(project1)
            If StrLen% < 10 Then
                  MsgType "Project # is inaccurate.  Try Again"
                  GoTo Try2
            End If
        Else
            project1 = "<ERASEEOF>"
      End If
Try3:
      project2 = ""
      If Ans2 = "No" Then
            MsgAsk  "Please input project number for 2nd CLLI (Press Enter when
                  finished):", project2, 10
            StrLen% = Len(project2)
            If StrLen% < 10 Then
                  MsgType "Project # is inaccurate.  Try Again"
                  GoTo Try3
            End If
            If skip = "yes" Then
                  GoTo Finish1
            End If
        Else
            project2 = "<ERASEEOF>"
      End If

      EMSetCursor 3,16
      If Ans1 = "No" Then
            EMSendKey "4.3"
        Else
            EMSendKey "4.5"
      End If
      EMSetCursor 5,8
      EMSendKey clli1
      EMSetCursor 5,31
      EMSendkey project1
      EMSetCursor 6,13
```

```
        EMSendkey "<ERASEEOF><ENTER>"
        SysDelay 2
        EMReadScreen finding,4,2,2
        If finding = "G009" Then
              MsgType "1st CLLI Code not valid"
              GoTo TryAgain
        End If
        If finding = "D003" Then
              MsgType "Must enter a project number for 1st CLLI office
                   conversion."
              Ans1 = "No"
              GoTo Try2
        End If
        If finding = "M048" Then
              MsgType "Project # is inaccurate.  Try Again"
              GoTo Try2
        End If
Nextline1:
        For i = 0 to 999
MoreLoop1:
        EMReadScreen finding,52,row,2
        tempDataArr1(i) = finding
        compare1(i) = left(finding,10)
        page1(i) = left(finding,7)
        If page1(i) = "More..." Then
              row = 15
              EMSendKey  "<PF8>"
              EMWaitCursor TimeOut ,3,16
              If NM_ResultCode = 2 Then
                    Goto HandelError
              End If
              GoTo MoreLoop1
        End If
        If page1(i) = "END     " Then
              row = 15
              GoTo Finish1
        End If
        EMReadScreen finding,1,row,59
        answer1(i) = finding
        row = row + 1
        Next i
Finish1:
        EMSetCursor 3,16
        If Ans2 = "No" Then
              EMSendKey "4.3"
          Else
              EMSendKey "4.5"
        End If
        EMSetCursor 5,8
        EMSendKey clli2
        EMSetCursor 5,31
        EMSendkey project2
        EMSetCursor 6,13
        EMSendkey "<ERASEEOF><ENTER>"
        SysDelay 2
        EMReadScreen finding,4,2,2
        If finding = "G009" Then
```

```
            MsgType "2nd CLLI Code not valid"
            GoTo TryAgain
      End If
      If finding = "D003" Then
            MsgType "Must enter a project number for 2nd CLLI office
                  conversion."
            Ans2 = "No"
            skip = "yes"
            GoTo Try3
      End If
      If finding = "M048" Then
            MsgType "Project # is inaccurate.  Try Again"
            skip = "yes"
            GoTo Try3
      End If
Nextline2:
      For i = 0 to 999
MoreLoop2:
      EMReadScreen finding,52,row,2
      tempDataArr2(i) = finding
      compare2(i) = left(finding,10)
      page2(i) = left(finding,7)
      If page2(i) = "More..." Then
            row = 15
            EMSendKey  "<PF8>"
            EMWaitCursor TimeOut ,3,16
            If NM_ResultCode = 2 Then
                  Goto HandelError
            End If
            GoTo MoreLoop2
      End If
      If page2(i) = "END     " Then
            row = 15
            GoTo Finish2
      End If
      EMReadScreen finding,1,row,59
      answer2(i) = finding
      row = row + 1
      Next i
Finish2:
      If Action =  "Same features but opposite status" Then
            Opposites
      End If
      If Action = "Same features and same status" Then
            Same
      End If
      If Action = "Differences between generics but not same features" Then
            Differences
      End If
      Stop
HandelError:
      Print "Error receiving text or cursor postion. Script terminated."
End Sub

Sub Opposites
      FileWrite "compare.txt", "Compare same feature packages with an
opposite
```

```
            status between offices"
      cllititle = "                                                    " & clli1 & "   " &
            clli2
      FileAppend cllititle
Same1:
      For i = 0 to 999
      If compare1(i) = "END       " Then
            GoTo Finish3
      End If
InSame1:
      For j = 0 to 999
      If compare2(i) = "END       " Then
            GoTo OutSame1
      End If
      If compare1(i) = compare2(j) Then
            If answer1(i) = answer2(j) Then
                  GoTo OutSame1
             Else
                  tempDataArr1(i) = tempDataArr1(i) & "       " & Answer1(i)
                        & "             " & Answer2(j)
                  FileAppend tempDataArr1(i)
                  GoTo OutSame1
            End If
      End If
      Next j
OutSame1:
      Next i
Finish3:
      Command = "NOTEPAD.EXE "
      Command = Command & "compare.txt"
      SysWinExec command
End Sub

Sub Same
      FileWrite "compare.txt", "Compare same feature packages with the same
            status between offices"
      cllititle = "                                                    " & clli1 & "   " &
            clli2
      FileAppend cllititle
Same2:
      For i = 0 to 999
      If compare1(i) = "END       " Then
            GoTo Finish4
      End If
InSame2:
      For j = 0 to 999
      If compare2(i) = "END       " Then
            GoTo OutSame2
      End If
      If compare1(i) = compare2(j) Then
            If answer1(i) = answer2(j) Then
                  tempDataArr1(i) = tempDataArr1(i) & "       " & Answer1(i)
                        & "             " & Answer2(j)
                  FileAppend tempDataArr1(i)
                  GoTo OutSame2
              Else
                  GoTo OutSame2
```

```
            End If
        End If
        Next j
OutSame2:
        Next i
Finish4:
        Command = "NOTEPAD.EXE "
        Command = Command & "compare.txt"
        SysWinExec command
End Sub

Sub Differences
        FileWrite "compare.txt", "Compare generic differences only eliminating
             common feature packages"
        cllititle = "                                                    " & clli1 & "   "
             & clli2
        FileAppend cllititle
Same3:
        For i = 0 to 999
        match = "N"
        If compare1(i) = "END        " Then
              GoTo Finish5
        End If
InSame3:
        For j = 0 to 999
        If compare2(i) = "END        " Then
              GoTo OutSame3
        End If
        If compare1(i) = compare2(j) Then
              match = "Y"
              GoTo OutSame3
        End If
        Next j
OutSame3:
        If Match = "N" Then
              tempDataArr1(i) = tempDataArr1(i) & "       " & Answer1(i)
              FileAppend tempDataArr1(i)
        End If
        Next i
Finish5:
Same4:
        For k = 0 to 999
        match = "N"
        If compare2(k) = "END        " Then
              GoTo Finish6
        End If
InSame4:
        For l = 0 to 999
        If compare1(k) = "END        " Then
              GoTo OutSame4
        End If
        If compare2(k) = compare1(l) Then
              match = "Y"
              GoTo OutSame4
        End If
        Next l
OutSame4:
```

```
      If Match = "N" Then
            tempDataArr2(k) = tempDataArr2(k) & "                    "
                  & Answer2(k)
            FileAppend tempDataArr2(k)
      End If
      Next k
Finish6:
      Command = "NOTEPAD.EXE "
      Command = Command & "compare.txt"
      SysWinExec command
End Sub
```